 Recommending News Articles to Unknown Users

**Description:**

You own a web site that random users access for news. Your goal is to choose a news article to show to each user, that will maximize the chance that the user clicks on it (to read it further). This is also known as "the clickthrough rate". This is your problem setup more formally:

* **News Articles:**
  + When a user arrives at your site, there are a total of 𝐾=5 news articles you can choose from.
  + If you recommend article 𝑖, then there is a probability that the user clicks article i, which is unknown and equal to 𝑝𝑖.
  + Assume you have a total of T rounds, during which you want to maximize the number of successful recommendations (i.e., clicks)
* **Users**:
  + For every user that visits your site, you know if they are: (i) male or female, and (ii) under or over 25 years old.
  + The "characteristics" of each new user visiting your site, are drawn in an IID manner (i.e., the next user has no dependence on who the previous user was).
  + For your simulations, assume initially that there's an equal probability to draw a user from any of these classes. You can later compare this also with one scenario where these probabilities are not equal. (In all cases, assume you don't know these probabilities, beforehand, either).
* **User-News Preference:**
  + Unlike the standard bandits we've seen, it turns out that different types of users might prefer different articles!
  + Let 𝑝0,𝑝1,𝑝2,𝑝3,𝑝4 denote the click probabilities for articles, 1,2,3,4,5, respectively. The taste differences are captured as follows:
    - female over 25: 𝑝0=0.8,𝑝1=0.6,𝑝2=0.5,𝑝3=0.4,𝑝4=0.2
    - male over 25: 𝑝0=0.2,𝑝1=0.4,𝑝2=0.5,𝑝3=0.6,𝑝4=0.8
    - male or female under 25:  𝑝0=0.2,𝑝1=0.4,𝑝2=0.8,𝑝3=0.6,𝑝4=0.5
  + **NOTE: Your algorithm initially knows NEITHER the ranking of different articles (per category), NOR the exact click probabilities. It doesn't even know that males and females under 25 have similar preferences**

**Tasks:**

1. Propose a simple modification of the UCB algorithm that (you believe) achieves sublinear regret for this problem, and code it in Python.
2. Prove an upper bound for the theoretical expected regret of the algorithm in this environment as a function of K (the arms), T (the horizon), and possibly the number of user types |𝑈| (this is equal to 4 in the above example). (**You don't need to come up with a brand new proof of your own! Just need to go through the steps of the UCB proof we did, and see which parts you might need to modify, to capture the slightly new environment)**
3. (For 𝑇=1000 ) Produce plots that (i) show the sublinear regret of your algorithm, (ii) compare the experimental regret you achieve with the theoretical upped bound you derived.
4. Repeat the experiment for  𝑇=10000 and compare to the previous one.